JOURNAL OF COMPUTER AND SYSTEM SCIENCES 6, 125–137 (1972)

# Equivalence of Programs with Structured Variables

A. V. AHO

*Bell Telephone Laboratories, Inc., Murray Hill, New Jersey*

AND

J. D. ULLMAN*

*Princeton University, Princeton, New Jersey*

Received January 22, 1971

We consider a class of straight line programs admitting structured variables. It is easy to associate with each program a set of expressions which reflects the natural meaning of a structured variable such as an array. However, the question of whether two such expressions are equivalent depends on what is assumed about the possible initial values of the variables and what algebraic laws are assumed to apply. We give necessary and sufficient conditions for two such assumptions to yield the same equivalences among expressions. The assumption which gives the smallest set of equivalences is exhibited. This assumption plays the role of "strong equivalence," since it implies equivalence under any possible interpretation of the model which preserves the interpretation of "structure" as we define it.

## 1. INTRODUCTION

We consider a class of program schemata that models straight line programs having both scalar and structured variables. Our goal is to consider the conditions under which we can use transformations to optimize programs containing structured variables and yet be assured that we are preserving program equivalence.

Variables representing vectors or arrays are examples of structured variables. Previous theoretical work on program optimization and equivalence has apparently shirked the case where structured variables are explicitly allowed [1, 4–6] or restricted the operations concerning the structured variables to localized optimizations of address computations of simple arrays in loops [2, 3].

In this paper we will take a general look at the problem of preserving equivalence

* This work was partially supported by NSF grant GJ-1052.

when attempting to transform programs with structured variables. We will concentrate on the difficulties that arise from the introduction of structured variables to programs. To avoid undecidability considerations [6], we will restrict ourselves to straight line programs.

We will use a model in which a fixed interpretation is placed on one operator (dot), which reflects structure in variables. We place arbitrary interpretations on "elementary expressions" (those in which the dot operator is applied only to initial values of arrays) and show that two interpretations define the same equivalence if and only if they define the same set of "dichotomies" on elementary expressions.

## 2. Programs

A program consists of a linear sequence of assignment statements which can be of three types. To describe statements formally we will use three types of symbols—operators, variables and metasymbols:

(1) $\Theta$ is the set of *operator symbols*. Each operator in $\Theta$ is $n$-ary for some $n \geqslant 1$;

(2) (a) $\mathscr{A}$ is a countable set of *scalar variable symbols*,

(b) $\mathscr{B}$ is a countable set of *structured variable symbols*;

(3) $\leftarrow$ and $\cdot$ are used as metasymbols.

As a general rule we will use $A, B, \ldots, J$ to denote scalar variables and $\alpha$, $\beta$ and $\gamma$ to denote structured variables.

Let $V$ be an arbitrary set of values. A structured variable $\alpha$ can be pictured as a mapping from $V$ into $V$. Each value in the domain of $\alpha$ can be construed as a "location" in $\alpha$. We write $\alpha \cdot A$ to denote the current value stored at the location in $\alpha$ given by the current value of $A$. Furthermore, we will assume that $\alpha \cdot A$ is defined for all values of $A$.

A *statement* is a string of symbols having one of the following three forms:

(1) $A \leftarrow \theta B_1 \cdots B_r$

(2) $A \leftarrow \alpha \cdot B$

(3) $\alpha \cdot A \leftarrow B$

A statement of the first type represents an instruction which assigns to the scalar variable $A$ the value obtained by applying operator $\theta$ to the current values of scalar variables $B_1, B_2, \ldots, B_r$.

A statement of the form $A \leftarrow \alpha \cdot B$ indicates the current value of $\alpha \cdot B$ is to be assigned to the scalar variable $A$.

A statement of the form $\alpha \cdot A \leftarrow B$ indicates that the current value of $B$ is to be assigned to the location in $\alpha$ given by the current value of $A$.

A *program* $\pi$ is a triple $(P, I, U)$ where $P$ is a sequence of statements separated by semicolons, and $I$ and $U$ are finite subsets of $\mathscr{A} \cup \mathscr{B}$, representing the input and output variables, respectively.

EXAMPLE 1. Let $\pi = (P, I, U)$ where $P$ is the sequence of statements

$$C \leftarrow \alpha \cdot A;$$
$$D \leftarrow \alpha \cdot B;$$
$$C \leftarrow *CD;$$
$$D \leftarrow +AB;$$
$$\alpha \cdot D \leftarrow C$$

$I = \{\alpha, A, B\}$ and $U = \{\alpha\}$. This program represents the computation $\alpha(A + B) = \alpha(A) * \alpha(B)$, treating the dot as a subscripting operator.

We will assume that in any program all variables which are referenced are either input variables or have been previously defined.

## 3. PROGRAM VALUE

*Expressions* (over $\mathscr{A}$, $\mathscr{B}$ and $\Theta$) can be scalar or structured and are defined as follows:

(1) $A$ in $\mathscr{A}$ is a scalar expression.

(2) $\alpha$ in $\mathscr{B}$ is a structured expression.

(3) If $E_1, \ldots, E_n$ are scalar expressions and $\theta$ is an $n$-ary operator, then $\theta[E_1][E_2] \cdots [E_n]$ is a scalar expression.

(4) If $E_1$ and $E_2$ are scalar expressions and $\epsilon$ is a structured expression, then $(E_1, E_2)\epsilon$ is a structured expression.

(5) If $\epsilon$ is a structured expression and $E$ a scalar expression, then $[\epsilon] \cdot [E]$ is a scalar expression.

(6) Nothing else is a scalar or structured expression.

(7) Square brackets may be deleted if no ambiguity arises.

We denote the set of scalar expressions by $\mathscr{E}$.

*Comment.* By (2) and (4), every structured expression is of the form

$$(E_1, V_1)(E_2, V_2) \cdots (E_n, V_n)\alpha,$$

where the $E$'s and $V$'s are scalar expressions and $\alpha$ is a structured variable. This expression is intended to represent the history of $\alpha$. In this history, the first assignment to $\alpha$ was $\alpha \cdot A \leftarrow B$, when the expressions for $A$ and $B$ were $E_n$ and $V_n$, respectively. The most recent assignment to $\alpha$ was $\alpha \cdot C \leftarrow D$ when the expressions for $C$ and $D$ were $E_1$ and $V_1$.

We can define precisely what we mean by an expression for a variable.

Let $\pi = (P, I, U)$ be a program with $P = S_1 ; S_2 ;...; S_m$. For $X$ in $\mathscr{A} \cup \mathscr{B}$ we define $e_t(X)$, the *expression for variable $X$ after time $t$*, as follows:

(1) $e_0(X) = X$ for all $X$ in $I$

(2) (i) If $S_t$ is $A \leftarrow \theta B_1 \cdots B_r$, then

$$e_t(A) = \theta[e_{t-1}(B_1)] \cdots [e_{t-1}(B_r)]$$

(ii) If $S_t$ is $A \leftarrow \alpha \cdot B$ then

$$e_t(A) = [e_{t-1}(\alpha)] \cdot [e_{t-1}(B)]$$

(iii) If $S_t$ is $\alpha \cdot A \leftarrow B$, then

$$e_t(\alpha) = (e_{t-1}(A), e_{t-1}(B))\, e_{t-1}(\alpha)$$

(iv) If $X$ is not set by $S_t$, then

$$e_t(X) = e_{t-1}(X)$$

provided $e_{t-1}(X)$ is defined.

The set of expressions for program $\pi$, denoted $e(\pi)$, is $\{e_m(X) \mid X \text{ is in } U\}$.

EXAMPLE 2. $e(\pi)$ for $\pi$ of Example 1 is $\{(+AB, *\alpha \cdot A\, \alpha \cdot B)\alpha\}$.

## 4. PROGRAM EQUIVALENCE

Informally, we will say that two scalar expressions are equivalent, under some interpretation, if there is no permissible assignment of values to the variables appearing in the expressions which will give the expressions different values under that interpretation. Two structured expressions $\epsilon_1$ and $\epsilon_2$ are equivalent if $\epsilon_1 \cdot E$ and $\epsilon_2 \cdot E$ are equivalent for all $E$ in $\mathscr{E}$.

Two programs $\pi_1$ and $\pi_2$ are equivalent if for each expression in $e(\pi_1)$ there is an equivalent expression in $e(\pi_2)$, and conversely. Thus the problem of determining program equivalence becomes one of determining the equivalence of expressions.

With structured variables available, determining when two expressions are equivalent is not an easy task. For example, it will not suffice to say, even in the most elementary circumstances, that two expressions are equivalent if and only if they are

identical. Let $\pi_1 = (P_1, \{\alpha, A, B\}, \{\alpha\})$ and $\pi_2 = (P_2, \{\alpha, A, B\}, \{\alpha\})$ be two programs with $P_1 = \alpha \cdot A \leftarrow B;\ \alpha \cdot B \leftarrow A$ and $P_2 = \alpha \cdot B \leftarrow A;\ \alpha \cdot A \leftarrow B$. We would expect that under any interpretation, $\pi_1$ and $\pi_2$ are equivalent programs, and hence, $(B, A)(A, B)\alpha$ and $(A, B)(B, A)\alpha$ are equivalent expressions.

To attack the problem of determining when two expressions are equivalent, we postulate the existence of a class of admissible "input settings" which give initial values to the scalar variables and to each location of the structured variables. In order to keep things as general as possible, but still allow us to formulate what we believe the dot operations mean, we will assume each input setting assigns a value to those expressions involving only the initial values of scalar and structured variables and the operators from $\Theta$. We will call such expressions "elementary." Each input setting induces an equivalence relation on elementary expressions; expressions are equivalent if and only if they are given the same value. Since we are only interested in equivalence of expressions, not their values, we will hereafter talk only about equivalence relations on expressions.

At this point, it is worthwhile to compare our intended definitions with the notion of "interpretation" found in [4–6]. An interpretation defines a space of values for variables and functions designated by the operators. The value of an expression is found by applying the appropriate functions in the obvious way. We define an interpretation on elementary expressions in exactly this sense. However, for our purposes the actual values of expressions are not important; it is sufficient to merely recognize when two expressions have the same value. Thus, we resort to equivalence relations which obscure the actual value, but have the same effect as some interpretation where equivalence of expressions is concerned.

As is usual, we say that two expressions or programs are equivalent if and only if they give the same values under an arbitrary interpretation (although in the literature, interpretations are sometimes restricted, e.g., to having recursive functions for the operators). However, we insist on a fixed interpretation of the dot, and it is in this assumption that our developments depart from previously trodden paths.

Formally, the set $\mathscr{E}_0 \subseteq \mathscr{E}$ of *elementary expressions* is defined recursively as follows:

(1) $A$ in $\mathscr{A}$ is an elementary expression.

(2) If $E_1, \ldots, E_n$ are elementary expressions and $\theta$ an $n$-ary operator, then $\theta[E_1][E_2] \cdots [E_n]$ is an elementary expression.

(3) If $\alpha$ is in $\mathscr{B}$, and $E$ is an elementary expression, then $\alpha \cdot E$ is an elementary expression.

(4) Nothing else is an elementary expression.

Our first goal is to make a reasonable extension of equivalence relations from $\mathscr{E}_0$ to the set of all scalar expressions, which have denoted $\mathscr{E}$. The key to such an extension is to find, for an equivalence relation $R$ on $\mathscr{E}_0$, and for each $E$ in $\mathscr{E}$, a canonical

elementary expression which we consider $R$-related to $E$. This can be done in the following manner.

Let $R$ be an equivalence relation on $\mathscr{E}_0$. For each $E$ in $\mathscr{E}$ we define $E'$, the *$R$-canonical elementary expression* for $E$, recursively as follows:

(1) If $E$ is an element of $\mathscr{E}_0$, then $E' = E$;

(2) If $E = \theta[E_1] \cdots [E_n]$, then $E' = \theta[E_1'] \cdots [E_n']$; where $E_i'$ is the $R$-canonical elementary expression for $E_i$, $1 \leqslant i \leqslant n$;

(3) Suppose $E = (E_1, V_1) \cdots (E_n, V_n)\, \alpha \cdot F$, $n \geqslant 0$. Let $i$ be the smallest integer such that $E_i' R F'$, where $E_i'$ and $F'$ are the $R$-canonical elementary expressions for $E_i$ and $F$. Then $E' = V_i'$, the $R$-canonical elementary expression for $V_i$. If there is no such $i$, then $E' = \alpha \cdot F'$. Thus, each input setting $R$ determines which location ($E_i$) is referred to by $F$ and the value at that location ($V_i$) is treated as the expression. Define $R^*$, *the extension of $R$ to $\mathscr{E}$*, by: $E R^* F$ if and only if $E' R F'$, where $E'$ and $F'$ are the $R$-canonical elementary expressions for $E$ and $F$.

Observe that if $E$ is an elementary expression, then $E' = E$. Thus, $R^*$ agrees with $R$ on $\mathscr{E}_0$. Also, $R^*$ is clearly an equivalence relation.

EXAMPLE 3. Suppose that $E_1, E_2, V_1, V_2, F$ and $G$ are elementary expressions and that $E_1 R F$ and $E_2 R G$. We assume no other pairs from $E_1, E_2, F$ and $G$ are $R$-related. Then the $R$-canonical elementary expression for both the expressions $(E_1, V_1)(E_2, V_2)\, \alpha \cdot F$ and $(E_1, V_2)(E_2, V_1)\, \alpha \cdot G$ is $V_1$. Thus it follows

$$(E_1, V_1)(E_2, V_2)\, \alpha \cdot F \; R^* (E_1, V_2)(E_2, V_1)\, \alpha \cdot G.$$

We are now in a position to say when two scalar expressions are equivalent under a set of relations (i.e., under a given interpretation).

Let $\mathscr{S}$ be a set of equivalence relations on $\mathscr{E}_0$, and let $E$ and $F$ be in $\mathscr{E}_0$. We say $E \equiv_{\mathscr{S}} F$ if and only if for all $R$ in $\mathscr{S}$, $E R F$. For each $E$ and $F$ in $\mathscr{E}$, we say $E \overset{*}{\equiv}_{\mathscr{S}} F$ if and only if for all $R$ in $\mathscr{S}$, $E R^* F$.

Note that $\equiv_{\mathscr{S}}$ and $\overset{*}{\equiv}_{\mathscr{S}}$ are equivalence relations and coincide on $\mathscr{E}_0$.

## 5. EQUIVALENT INTERPRETATIONS ON $\mathscr{E}_0$

The question we shall now investigate is under what circumstances two sets of equivalence relations (interpretations) on $\mathscr{E}_0$ induce the same $\overset{*}{\equiv}$ relations on $\mathscr{E}$. It is not sufficient that they induce the same $\equiv$ relations on $\mathscr{E}_0$.

EXAMPLE 4. Let $\mathscr{S}_0 = \{R_0\}$, where $R_0$ is the identity relation on $\mathscr{E}_0$ (i.e., $E R_0 F$ iff $E = F$). Then $\overset{*}{\equiv}_{\mathscr{S}_0}$ is also the identity relation on $\mathscr{E}_0$. Let $\mathscr{S}$ consist of relations $R_E$

for all $E$ in $\mathscr{E}_0$ defined by $F\,R_E\,G$ if and only if both or neither of $F$ and $G$ are $E$. Then $F\,R_F\,G$ is false if $F \neq G$. Thus $\equiv_{\mathscr{S}} = \equiv_{\mathscr{S}_0}$.

However, let $E$, $F$, $V$ and $W$ be four distinct elementary expressions, and consider the expressions $G_1 = (E, V)\,\alpha \cdot E$ and $G_2 = (F, W)(E, V)\,\alpha \cdot E$. Then $G_1 \overset{*}{\equiv}_{\mathscr{S}_0} G_2$, because $E\,R_0\,F$ is false and the $R_0$ canonical expressions for $G_1$ and $G_2$ are both $V$. We claim that $G_1 \overset{*}{\equiv}_{\mathscr{S}} G_2$ is false because, for example, the $R_V$-canonical expression for $G_1$ is $V$ and that for $G_2$ is $W$, since $E\,R_V\,F$. Since $V\,R_V\,W$ is false, it follows that $G_1\,R_V\,G_2$ is false, and hence $G_1 \overset{*}{\equiv}_{\mathscr{S}} G_2$ is false.

### 5.1. *Dichotomies*

We will now derive the exact conditions under which two sets of equivalence relations on $\mathscr{E}_0$ induce the same $\overset{*}{\equiv}$ relations on $\mathscr{E}$. These conditions can be readily expressed in terms of a set of finite dichotomies on expressions which are induced by the equivalence relations. Informally, a dichotomy is a picture of the effect of an input setting on a finite set of expressions, listing some pairs of expressions which acquire the same value and pairs which do not acquire the same value.

A *dichotomy* is a pair $(M, N)$, where

(1) $M$ and $N$ are both finite subsets of $\mathscr{E} \times \mathscr{E}$, and

(2) $N$ is nonempty.

A dichotomy $(M, N)$ is *satisfied* by an equivalence relation $R$ if

(3) For all $(E, F)$ in $M$, $E\,R^*\,F$, and

(4) For all $(G, H)$ in $N$, $G\,R^*\,H$ is false.

For example, $(\{(A, A)\}, \{(A, B)\})$, where $A \neq B$, is a dichotomy satisfied by $R_0$, the identity relation.

Let $\mathscr{S}$ be a set of equivalence relations. The *set of dichotomies associated with* $\mathscr{S}$, denoted $D(\mathscr{S})$, is the set of dichotomies satisfied by $R$ for some $R$ in $\mathscr{S}$.

The *elementary dichotomies associated with* $\mathscr{S}$, denoted $D_0(\mathscr{S})$, is a set defined in the same way as $D(\mathscr{S})$, except in condition (1), $M$ and $N$ are restricted to be finite sets of pairs in $\mathscr{E}_0 \times \mathscr{E}_0$.

EXAMPLE 5. Let $\mathscr{S}_0$ and $\mathscr{S}$ be as in Example 4. Then $D_0(\mathscr{S}_0)$ is the set of pairs $(M, N)$ such that if $(E, F)$ is in $M$, then $E = F$, and if $(G, H)$ is in $N$, then $G \neq H$. Note that $N$ must not be empty, by definition of "dichotomy."

$D_0(\mathscr{S})$ is the set of pairs $(M, N)$ such that

(1) $N$ contains no pair of the form $(F, F)$ and

(2) there exists $E$ in $\mathscr{E}_0$ such that $E$ is in every pair of $N$ and each pair of $M$ either is $(E, E)$ or omits $E$.

To see this, suppose $(M, N)$ is a dichotomy satisfied by some $R_E$ in $\mathscr{S}$. Then for every $(F, G)$ in $N, F \neq G$, and either $F = E$ or $G = E$ (since only then will $F\, R_E\, G$ be false). If $(H, J)$ is in $M$, then $H\, R_E\, J$. Hence either $H = E = J$ or $H \neq E$ and $J \neq E$.

We observe that $D_0(\mathscr{S}_0) \neq D_0(\mathscr{S})$, and in fact, these two sets are incommensurate.

## 5.2. *The Main Result*

LEMMA 1. *Let $\mathscr{S}_1$ and $\mathscr{S}_2$ be sets of equivalence relations. Then $D(\mathscr{S}_1) \subseteq D(\mathscr{S}_2)$ if and only if $D_0(\mathscr{S}_1) \subseteq D_0(\mathscr{S}_2)$.*

*Proof.* The "only if" portion is trivial. For the "if" portion, suppose that $D_0(\mathscr{S}_1) \subseteq D_0(\mathscr{S}_2)$, but there exists a dichotomy $(M, N)$ in $D(\mathscr{S}_1)$ and not in $D(\mathscr{S}_2)$. Let

$$M = \{(E_1, F_1), \ldots, (E_m, F_m)\} \qquad \text{and} \qquad N = \{(G_1, H_1), \ldots, (G_n, H_n)\}.$$

Then there is some $R$ in $\mathscr{S}_1$ which satisfies $(M, N)$. We will show that from such a dichotomy we can construct by a sequence of reductions a dichotomy which lies in $D_0(\mathscr{S}_1) - D_0(\mathscr{S}_2)$ and which is satisfied by $R$.

Let $E$ be any expression appearing in $M$ or $N$ which has a subexpression of the form $(I_1, V_1) \cdots (I_k, V_k)\, \alpha \cdot J$, where $k \geqslant 1$. Since $(M, N)$ cannot be in $D_0(\mathscr{S}_1)$, there must exist such an $E$. From $(M, N)$ form $(M', N')$, a *reduction* of $(M, N)$, as follows:

(1) Let $i$ be the smallest integer such that $I_i\, R^*\, J$, or let $i = k + 1$ if none exists. Replace the mentioned subexpression of $E$ by $V_i$ if $i \leqslant k$, or $\alpha \cdot J$ if $i = k + 1$, and call the resulting expression $E'$.

(2) Add $(I_i, J)$ to $M$ if $i \leqslant k$, and add nothing to $M$ if $i = k + 1$. Call the result $M'$.

(3) Add $(I_1, J), (I_2, J), \ldots, (I_{i-1}, J)$ to $N$. Call the result $N'$.

Then $E\, R^*\, E'$, since it is easy to see that they have the same $R$-canonical elementary expression. Also, $I_i\, R^*\, J$ if $i \leqslant k$, but $I_j\, R^*\, J$ is false for $1 \leqslant j < i$. Hence, $(M', N')$ is satisfied by $R$.

Suppose there exists $T$ in $\mathscr{S}_2$ satisfying $(M', N')$. It follows, since $I_j\, T^*\, J$ is false for $j < i$, but $I_i\, T^*\, J$, if $i \leqslant k$, that $E$ and $E'$ have the same $T$-canonical elementary expression, and hence, $E\, T^*\, E'$. Thus, $(M, N)$ would be satisfied by $T$. Therefore, we conclude that $(M', N')$ is in $D(\mathscr{S}_1) - D(\mathscr{S}_2)$.

All that remains is to show that for any $(M, N)$ in $D(\mathscr{S}_1) - D(\mathscr{S}_2)$, there is some sequence of reductions that will yield $(M_0, N_0)$ in $D_0(\mathscr{S}_1) - D_0(\mathscr{S}_2)$. Since there is no such $(M_0, N_0)$, we will have the lemma.

The choice of $E$ should be from among those nonelementary expressions of $M$ and

$N$ having the largest number of dots. Since none of the expressions $E', I_j, 1 \leqslant j \leqslant i$ and $J$ has as many dots as $E$, each reduction reduces either

(1) the number of nonelementary expressions in $M$ and $N$ having the maximum number of dots (without increasing the maximum number of dots), or

(2) the maximum number of dots found in any one of these nonelementary expressions.

We cannot reduce (1) indefinitely, and can never increase (2), so after a finite number of reductions of type (1), a reduction of type (2) will take place. Since we cannot reduce (2) indefinitely, we will eventually obtain a dichotomy in $D_0(\mathscr{S}_1) - D_0(\mathscr{S}_2)$.

LEMMA 2. *Let $E, F, G$ and $H$ be in $\mathscr{E}$, $\alpha$ in $\mathscr{B}$ and let $R$ be an equivalence relation. Then*:

(a) *$G\,R^*\,(F, H)(E, G)\,\alpha \cdot E$ is false if and only if $E\,R^*\,F$ is true and $G\,R^*\,H$ is false.*

(b) *$G\,R^*\,(F, G)(E, H)\,\alpha \cdot E$ is false if and only if both $E\,R^*\,F$ and $G\,R^*\,H$ are false.*

*Proof.* For both (a) and (b), it suffices to consider the four possibilities of truth and falsehood for $E\,R^*\,F$ and $G\,R^*\,H$, and compute the $R$-canonical elementary expressions in each case.

The following lemma shows that the larger the set of dichotomies, the more inequivalent expressions we find.

LEMMA 3. *Let $\mathscr{S}_1$ and $\mathscr{S}_2$ be sets of equivalence relations on $\mathscr{E}_0$. Then $\overset{*}{\equiv}_{\mathscr{S}_1} \subseteq \overset{*}{\equiv}_{\mathscr{S}_2}$ if and only if $D(\mathscr{S}_2) \subseteq D(\mathscr{S}_1)$.*

*Proof. If*: Suppose $E \overset{*}{\equiv}_{\mathscr{S}_1} F$. Then the dichotomy $(\varphi, \{(E, F)\})$ is not in $D(\mathscr{S}_1)$. Hence it is also not in $D(\mathscr{S}_2)$, so $E \overset{*}{\equiv}_{\mathscr{S}_2} F$.

*Only if*: Suppose that $\overset{*}{\equiv}_{\mathscr{S}_1} \subseteq \overset{*}{\equiv}_{\mathscr{S}_2}$, but there exists a dichotomy $(M, N)$ in $D(\mathscr{S}_2) - D(\mathscr{S}_1)$. We construct another dichotomy $(\varphi, N_1)$ in $D(\mathscr{S}_2) - D(\mathscr{S}_1)$ as follows. If $M = \varphi$, then $N_1 = N$. Otherwise, let $(E, F)$ be in $M$ and $(G, H)$ be in $N$. Delete these and add $(G, (F, H)(E, G)\,\alpha \cdot E)$ to $N$ for some $\alpha$ in $\mathscr{B}$. By Lemma 2(a), the resulting dichotomy is satisfied by a relation $R$ if and only if the original one was. Repeating this transformation once for each element of $M$ yields $(\varphi, N_1)$ in $D(\mathscr{S}_2) - D(\mathscr{S}_1)$.

Next, we construct from $(\varphi, N_1)$ a dichotomy $(\varphi, N_2)$ in $D(\mathscr{S}_2) - D(\mathscr{S}_1)$, where $N_2$ is a singleton. If $N_1$ has a single element, we are done. Otherwise, let $(E, F)$ and $(G, H)$ be in $N_1$. Replace these by $(G, (F, G)(E, H)\,\alpha \cdot E)$. By Lemma 2(b), the new dichotomy is in $D(\mathscr{S}_2) - D(\mathscr{S}_1)$. Repeat this step one fewer times than there are elements of $N_1$, and the desired $(\varphi, N_2)$ will result.

Let $N_2 = \{(E, F)\}$. Since $(\varphi, N_2)$ is in $D(\mathscr{S}_2)$, $E \overset{*}{\equiv}_{\mathscr{S}_2} F$ is false. But since it is not in $D(\mathscr{S}_1)$, we have $E \overset{*}{\equiv}_{\mathscr{S}_1} F$, contradicting the assumption that $\overset{*}{\equiv}_{\mathscr{S}_1} \subseteq \overset{*}{\equiv}_{\mathscr{S}_2}$.

MAIN THEOREM. *The following statements about sets $\mathscr{S}_1$ and $\mathscr{S}_2$ of relations on $\mathscr{E}_0$ are equivalent*:

(1) $\overset{*}{\equiv}_{\mathscr{S}_1} \subseteq \overset{*}{\equiv}_{\mathscr{S}_2}$,

(2) $D_0(\mathscr{S}_2) \subseteq D_0(\mathscr{S}_1)$,

(3) $D(\mathscr{S}_2) \subseteq D(\mathscr{S}_1)$.

*Proof.* (2) is equivalent to (3) by Lemma 1 and (1) is equivalent to (3) by Lemma 3.

COROLLARY 1. $\overset{*}{\equiv}_{\mathscr{S}_1} = \overset{*}{\equiv}_{\mathscr{S}_2}$ *if and only if* $D_0(\mathscr{S}_1) = D_0(\mathscr{S}_2)$.

COROLLARY 2. *Let $\mathscr{S}_\infty$ be the set of all equivalence relations on $\mathscr{E}_0$. Then if $\mathscr{S}$ is any set of equivalence relations on $\mathscr{E}_0$, $\overset{*}{\equiv}_{\mathscr{S}_\infty} \subseteq \overset{*}{\equiv}_{\mathscr{S}}$.*

## 6. TRANSFORMATIONS ON PROGRAMS

Suppose that we would like to optimize a program containing structured variables by applying certain optimizing transformations to it. At all times we wish the result of any such optimization to be an equivalent program.

By Corollary 2, one "safe" way to modify expressions (or the programs which compute them) is to perform only those transformations which preserve $\overset{*}{\equiv}_{\mathscr{S}_\infty}$. Obviously, any set of transformations which does not preserve $\overset{*}{\equiv}_{\mathscr{S}_\infty}$ will not be useful for an arbitrary set of relations on $\mathscr{E}_0$. Thus those transformations that preserve $\overset{*}{\equiv}_{\mathscr{S}_\infty}$ are the only transformations that will be valid independent of the actual interpretation.[1] Thus $\overset{*}{\equiv}_{\mathscr{S}_\infty}$ plays the role played by strong (under all interpretations) equivalence in program schemata without structured variables [1].

It might be convenient if there were some simple set $\mathscr{S}$ such that $D_0(\mathscr{S}) = D_0(\mathscr{S}_\infty)$. This situation might make computation of equivalences among expressions simple. It is unfortunate that the following is true.

THEOREM. *If $\mathscr{S}$ is any set of relations such that $D_0(\mathscr{S}) = D_0(\mathscr{S}_\infty)$, then there is another set $\mathscr{S}'$ such that $\mathscr{S}' \subseteq \mathscr{S}$, $\mathscr{S}' \neq \mathscr{S}$, and $D_0(\mathscr{S}') = D_0(\mathscr{S}) = D_0(\mathscr{S}_\infty)$.*

[1] Strictly speaking, we might reasonably restrict the equivalence relations in some way. For example, we might insist that all relations $R$ have the property that if $E$ is an expression and $E'$ is formed from $E$ by substituting for some subexpression of $E$ an $R$-related expression, then $E\,R\,E'$. Thus, the dichotomy $(\{(X, Y)\}, \{(+XZ, +YZ)\}$ could not be satisfied. If we do so, then we get a different $\mathscr{S}_\infty$. The main assertion of this paper is that one makes all assumptions about the relations at one's own risk.

*Proof.* Let $\mathscr{S}$ be any such set and $R$ in $\mathscr{S}$. We claim that $D_0(\mathscr{S} - \{R\}) = D_0(\mathscr{S}_\infty)$. Otherwise, there is a dichotomy $(M, N)$ in $D_0(\mathscr{S})$ which is satisfied by $R$ but by no other member of $\mathscr{S}$. Let $E$ and $F$ be expressions not appearing in $M$ or $N$. If $E\,R\,F$, then let $M' = M$ and $N' = N \cup \{(E, F)\}$. If $E\,R\,F$ is false, let $M' = M \cup \{(E, F)\}$ and $N' = N$. In either case, $R$ does not satisfy $(M', N')$, so there exists $R'$ in $\mathscr{S}$ which does. (This is because there clearly exists some equivalence relation which satisfies $(M', N')$.) $R'$ also satisfies $(M, N)$, in contradiction to our assumption about that dichotomy.

COROLLARY. *There are no finite sets $\mathscr{S}$ such that $D_0(\mathscr{S}) = D_0(\mathscr{S}_\infty)$.*

## 7. CONCLUSIONS

We have found necessary and sufficient conditions that one set of assumptions about data and operators yields equivalences of expressions which are properly contained in those from some other assumption. Perhaps the most important application is the following apparent paradox.

Let $\mathscr{S}$ be a set of equivalence relations on elementary expressions, and let $\mathscr{S}'$ be obtained from $\mathscr{S}$ by making some algebraic law hold. To be specific, assume $+$ is in $\Theta$ and that $\mathscr{S}'$ is formed from $\mathscr{S}$ by making $+$ commutative. That is, we replace each $R$ in $\mathscr{S}$ by $R'$, the least equivalence relation containing $R$ such that $+GH\,R'\,+HG$ for all $G$ and $H$ in $\mathscr{E}_0$.

One would think that $\overset{*}{\equiv}_{\mathscr{S}'}$ would include $\overset{*}{\equiv}_{\mathscr{S}}$. That is, expressions which were equivalent before $+$ "became" commutative could not become inequivalent. However, the main theorem tells us that $\overset{*}{\equiv}_{\mathscr{S}} \subseteq \overset{*}{\equiv}_{\mathscr{S}'}$ if and only if $D_0(\mathscr{S}') \subseteq D_0(\mathscr{S})$. Is it possible that this latter inclusion does not hold?

Consider the dichotomy $(\{(+AB, +BA)\}, \{(C, D)\})$, where $A, B, C$ and $D$ are in $\mathscr{A}$. It is possible that this dichotomy is not in $D_0(\mathscr{S})$. That is, there is no reason why $+AB\,R\,+BA$ should be true for *any* $R$. However, it is entirely possible that there is an $R$ in $\mathscr{S}$ such that $C\,R\,D$ is false. If so, then $C\,R'\,D$ is also false. However, $+AB\,R'\,+BA$ for *every* $R$. Hence the dichotomy $(\{(+AB, +BA)\}, \{(C, D)\})$ may very well be in $D_0(\mathscr{S}') - D_0(\mathscr{S})$. Thus we conclude that $\overset{*}{\equiv}_{\mathscr{S}} \subseteq \overset{*}{\equiv}_{\mathscr{S}'}$ is not always true.

Translating this dichotomy into a related program, we might have the following sequence of statements:

$$E \leftarrow +AB$$

$$F \leftarrow +BA$$

$$\alpha \cdot E \leftarrow C$$

$$\alpha \cdot F \leftarrow D$$

The final value of $\alpha$ is $(+BA, D)(+AB, C)\alpha$. It is possible that $\mathscr{S}$ is such that $C \overset{*}{\equiv}_{\mathscr{S}} [(+BA, D)(+AB, C)\alpha] \cdot [+AB]$. It is sufficient, for example, that $+BA\ R\ +AB$ be false for all $R$ in $\mathscr{S}$. However, unless $D\ R\ C$ for all $R$ in $\mathscr{S}$ we cannot have

$$C \overset{*}{\equiv}_{\mathscr{S}} [(+BA, D)(+AB, C)\alpha] \cdot [+AB].$$

As a result, for certain data spaces and operators, possibly pathological ones, not "knowing" about the commutative law (or any other algebraic law) for some operator may lead to the fatal error of transforming one expression or program into an inequivalent one without realizing it.

The moral, if there is one, is that when dealing with structured variables, one cannot be too careful about what one assumes or does not assume concerning the ambient algebra.

## 8. Open Questions

The following open questions suggest themselves:

(1) Given set $\mathscr{S}$ of equivalence relations on $\mathscr{E}_0$, characterize the transformations on programs and/or expressions which preserve $\overset{*}{\equiv}_{\mathscr{S}}$. Most important is the case where $\mathscr{S}$ is $\mathscr{S}_\infty$ or the set of all equivalence relations $R$ such that the substitution of $R$-related expressions yields $R$-related expressions.

(2) Find a set of constraints on sets $\mathscr{S}$ of relations so that the "paradox" mentioned in Section 7 cannot occur, yet the usual sets of relations (those that reflect integer or binary arithmetic on arrays, for example) qualify.

## Acknowledgments

The authors would like to thank Brian Kernighan and Douglas McIlroy for their comments on this manuscript.

## References

1. A. V. Aho and J. D. Ullman, Transformations on straight line programs, pp. 136–148, *Conf. Record, Second Annual ACM Symposium on Theory of Computing*, May, 1970.
2. F. E. Allen, Program optimization, *in* "Annual Review in Automatic Programming," Vol. 5, Pergamon, New York, 1969.
3. John Cocke and T. J. Schwartz, "Programming Languages and Their Compilers," Courant Institute of Mathematical Sciences, 1969.

4. D. C. COOPER, Program scheme equivalences and second-order logic, *in* "Machine Intelligence 4" (Meltzer and Michie, Eds.), pp. 3–15, Edinburgh Univ. Press, Edinburgh, 1969.
5. D. M. KAPLAN, Proving things about programs, *Proc. Fourth Annual Princeton Conf. on Information Sciences and Systems*, pp. 244–251, March 1970.
6. D. LUCKHAM, D. PARK, AND M. S. PATERSON, On formalized computer programs, *J. Comput. System Sci.* **4** (1970), 205–219.